The Neighborhood Covering Heuristic (NCH) Approach for the General Mixed Integer Programming Problem

Final Phase I Report

Anthony Sterns, Ph. D.
Principal Investigator
Vice President, Research
Creative Action LLC
drtone@gwis.com
330 867 9978 ext 110

Ronni Sterns, Ph. D.
Corporate Officer
President and CEO
Creative Action LLC
rsterns@gwis.com
330 867 9978 ext 100

Jeffrey Adler, Ph. D.
Research Institution Principal Investigator
Assistant Professor
Department of Theoretical and Applied Mathematics
The University of Akron
adler@uakron.edu
330 972 6779

Douglas Kline, Ph. D.
Optimization Consultant
Assistant Professor
Information Systems and Operations Management
The University of North Carolina, Wilmington
klined@uncw.edu
910 962 7552

Scott Collins
Software Engineering and Computer Science Consultant
Chief Scientist
Ann Arbor Digital Arts
scc@scottcollins.net
734 327 9660

STTR Topic No.: N03-T004
Optimizing Human Resource Management Models

Phase I Contract No.: N00014-03-M-0254

July 1, 2003 to January 31, 2004

Cost Type Government Contract
No DCAA Audit to Date

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
02 FEB 2004	**Final**	**01 Jul 2003 - 02 Feb 2004**

4. TITLE AND SUBTITLE
The Neighborhood Covering Heuristic (NCH) Approach for the General Mixed Integer Programming Problem

5a. CONTRACT NUMBER
N00014-03-M-0254

5b. GRANT NUMBER
N03-T004

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Anthony Sterns, Ph.D., Jeffrey Adler, Ph.D., Douglas Kline, Ph.D., and Scott Collins.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Creative Action LLC 680 N. Portage Path Akron, OH 44303; The University of Akron Department of Theoretical and Applied Mathematics Akron OH 44325-4002

8. PERFORMING ORGANIZATION REPORT NUMBER
SF309

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research Attn: Tanja Blackstone, Ph. D. Ballston Tower One 800 Quincy Street Arlington, VA 22217-5660; STTR Program Coordinator Attn: John Williams Code 364 Office of Naval Research 800 North Quincy Street Arlington, VA 22217-5660

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
N00014-03-M-0254

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
The research objectives were achieved and exceeded., The original document contains color images.

14. ABSTRACT
We accomplished our objectives, successfully implementing the Neighborhood Covering Heuristic (NCH) for solving the mixed integer programming problems. NCH is a unique, proprietary approach with several ground-breaking advantages. We completed a series of comparisons between NCH and the standard Branch and Bound (BAB) approach. Using the tunable parameters available within NCH, we created ten variants. We randomly generated sets of MIP problems, where the numbers of integer and binary variables, and constraints were varied in a systematic way. Then we applied both BAB and our ten variants of NCH to each of the generated problems. We obtained several significant results. First, when other parameters are fixed, the number of integer and binary variables in a random MIP problem does not have an exponential impact on the time required to find a feasible solution using NCH. Second, the performance data show that NCH produces feasible solutions significantly faster than BAB. Moreover the variance in time to produce a feasible solution is smaller in NCH. This reduction in variance and the resulting greater predictability of time to first solution will be extremely attractive to logistic companies, consultants, and useful directly in the Navy COMPASS program specifically and for Navy optimzation needs in general.

15. SUBJECT TERMS
STTR Report Optimization Mixed Integer Programming Problems Human Resources

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**31**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Contents

List of Figures

List of Tables

1. Identification and Significance of the Problem

1.1. **Significance.** Many real-world decisions can be cast as mixed integer programming (MIP) problems, including resource allocation, blending, and scheduling. Commonly known special cases of the general MIP problem are the knapsack problem, the assignment problem, the lockbox problem, the network flow problem, and the set-covering problem. The application areas for techniques for solving the MIP problem include human resource planning, financial planning, manufacturing, distribution, inventory, airline scheduling, telecommunications network planning, and many others. All of these disciplines in both the military and commercial workplace require computer software to aid in solving these complex but common business challenges.

The Office of Naval Research (ONR) is developing its core investment program, the Capable Manpower FNC IPT, which is focused on identifying and filling capability gaps, fulfilling commitments to funded acquisition programs, and designing a strategy that would provide the ability to execute the program. Each enabling capability has a set of milestones and transition opportunities. A solution method for the mixed integer problem, proposed here, would have tremendous value to both the private sector and ONR, specifically to utilizing capable manpower to the fullest.

Within the larger systems of Navy manpower and personnel supply, MIP problems are used in recruiting, selection and classification, training, and distribution and assignment. Recruiting problems involve allocation of resources such as signing bonuses and other programs to meet goals within resource constraints. Selection and classification problems involve meeting manpower needs while maximizing the fit of personnel to area. Training problems involve scheduling and allocating resources to training while minimizing costs. Distribution and assignment must match sailors to positions and accomplish the relocation of those sailors.

Providing an analytical tool that allows available personnel to be optimally assigned to maximize their interest and satisfy fleet requirements within practical real world time constraints will, if successful, have a significant impact towards the accomplishment of ONR goals and long-term personnel initiatives. This tool will improve the understanding of operational environments and give personnel the ability to operate under diverse, challenging conditions.

1.2. **Problem definition.** The general Mixed-Integer Programming (MIP) problem is stated in standard form as

$$\begin{array}{lll} \text{Maximize} & z = c \cdot x & (1) \\ \text{subject to} & Ax = b & (2) \\ \text{where} & x_i \geq 0 \quad \text{for all } i & (3) \\ \text{and} & x_i \text{ is integral, for all } i \text{ in a particular index set of size } d > 0. & (4) \end{array}$$

In the above, c is a $1 \times n$ vector representing costs, x is an $n \times 1$ vector of decision variables, A is an $m \times n$ matrix, and b is an $m \times 1$ vector. The matrix A is the constraint matrix. Note that the MIP can also be stated as a minimization problem, and that the equality in (2) may be created from inequalities by the addition of slack or surplus variables. Techniques for solving Linear Programming (LP) problems, defined by only lines (1)–(3), are well established. However, there is no general method for quickly determining the solution of a general MIP problem. The addition of (4) complicates matters, since the feasible solution space is no longer convex, but is broken into many non-connected spaces that must be searched. The number of spaces that must be searched is exponentially related to d. Integer variables may be further constrained to values less than or equal to one, and greater than or equal to zero. This effectively constrains integer variables to values of zero or one, i.e., binary.

1.3. **Existing techniques.** The traditional technique for solving the MIP problem is called the Branch-and-Bound (BAB) method. The BAB method involves solving a series of relaxed versions of the MIP problem, where all decision variables can take on real values, i.e., the integrality constraints are relaxed. Each relaxed problem is a standard LP problem, for which very good solution methods exist. If the relaxed problem's solution contains decision variable values that violate the integer-value constraint, then sub-problems (branches) are formed with additional constraints.

In situations where d is large, there can be many branches, and thus many sub-problems are generated. In the worst case, more than 2^{d+1} sub-problems may need to be solved. Each sub-problem is an LP problem. At each solution of a relaxed problem two choices must be made: (i) which decision variable to branch on and ii) which branch to follow. The basis of these decisions to date has been heuristic in nature, and no single heuristic has been shown to perform better than others on all problems.

The heuristics for choosing the variable on which to branch fall into two categories: deterministic and non-deterministic. Deterministic heuristics will choose the same variable to branch on each time they are run. A simple

deterministic heuristic would be to choose the "first" variable, i.e., based on an arbitrary predetermined variable index. A commonly used deterministic heuristic is to choose the variable that has the greatest impact on the objective function. This is intuitively appealing, but has not been shown to be consistently better than other heuristics.

Unlike linear programming problems, MIP problems have a large number of disjoint feasible regions that must be searched. Because the MIP problem is effectively a large-scale search that confounds local optimization methods, global search heuristics have been used to guide the BAB method. These search heuristics include pure random choice of variable, genetic algorithms (Goldberg, 1989 and 2002; Knjazew, 2001), and Tabu search (Glover and Laguna, 1997). These are all non-deterministic methods, since separate runs of these methods may choose different variables on which to branch.

It should be noted that all of the above are methods for making the necessary choices in the BAB method. Once the branching variable and the direction are chosen, each branch still produces a linear programming problem that must be solved, and may produce further branches.

Several drawbacks of the BAB method should be noted:

(1) **Solving sub-problems is computationally expensive.** There is significant computational effort spent in formulating and solving each sub-problem.
(2) **The number of sub-problems is exponential.** As stated above, there are potentially more than 2^{d+1} sub-problems. The BAB method is effectively a combinatorial search problem. In combination with (1), this can confound solution of the problem in a reasonable period of time.
(3) **One risks never getting a useful solution.** The initial solution to the relaxed problem is not necessarily feasible, and those of sub-problems may not be feasible either. Thus, early stopping of the algorithm may not produce a feasible solution.

2. Phase I Technical Objectives

Beyond the implicit objective of implementing the Neighborhood Covering Heuristic (NCH) for solving the mixed-integer programming problem, we had two objectives:

(1) Study different variants of NCH afforded by its tunability.
(2) Complete a series of statistically rigorous comparisons between NCH and the standard Branch and Bound (BAB) approach.

We achieved these objectives, and several others as well:

(3) Use NCH to solve several real-world problems.
(4) Use NCH to solve many large problems.

3. Methodology

In order to study variants of NCH and to compare them with BAB, we first implemented all of these methods in software, and then applied them to a collection of randomly generated problems.

3.1. **Platform.** Our prototype system was developed using MATLAB, a well-established platform for technical computing, which is published by The Mathworks (`http://www.mathworks.com`). MATLAB was well suited for our purposes, since it features easy desktop development and testing, a high-level scripting language, and a specialized optimization toolkit. With MATLAB, our development and simulations could be performed on standard desktop computers.

To clearly compare the NCH against existing methods such as BAB, the supporting methods, such as linear program solving and matrix manipulation, were done using the MATLAB Optimization Toolbox. In this way, we were able to clearly identify performance characteristics attributable to our method, rather than to specific code implementations of the supporting methods.

3.2. **Generation of Problems.** To perform a repeatable, statistically rigorous study, we needed to generate a set of problems. These problems must, of course, have feasible solutions. Furthermore, we must be able to control the characteristics of the problems and create multiple, different problems with the same set of characteristics. To accomplish this, we devised a method for generating MIP problems using the pseudo-random number generator in MATLAB. We fully describe our method here.

The basic idea is to create a point, and generate constraints around that point. In this way, we assure that this point is a feasible solution to the generated problem, i.e., we guarantee at least one feasible solution.

3.2.1. *Inputs.*

- `seed`: the seed for the random number generator
- `num_vars`: the number of variables in the problem
- `num_constraints`: the number of constraints in the problem
- `density_range_lo` and `density_range_hi`: specify about how many values in the constraint matrix A are nonzero
- `pct_integer`: specifies about how many variables are to be integer-valued
- `pct_binary`: specifies about how many variables are to be binary-valued.

3.2.2. *Output.*

- `prob`: a single mixed-integer programming problem.

A preliminary note is in order on the parameters used for generating random numbers. The generation of these problems is relatively sensitive to the parameters (means, standard deviations, etc.) used. The parameters used generate reasonable problems. They were determined by initial common sense, and trial and error.

Here are the basic steps:

3.2.3. *Generate the objective function.* The problems are to be minimization problems. We generate a vector of length `num_vars` from a Normal distribution with mean -200 and variance of 20. In general, this should guarantee that each variable will have an effect on the objective function, and all will be negative. Note that the objective function value does not in any way depend on which variables are real-valued, integer-valued, or binary-valued.

3.2.4. *Decide which variables are to be integer and binary.* The parameter `pct_integer` specifies a rough percentage of the `num_vars` variables that will be integer-valued. We ensure that at least one variable is integer-valued. To choose a random set of variables, a vector of Uniform$(0, 1)$ values of length `num_vars` is generated. Variables corresponding to values less than `pct_integer` are chosen to be integer-valued. If necessary, repeat until at least one variable is integer-valued.

Binary variables are generated in the same manner, except that we enforce that if a variable has been set as integer-valued, we do not change it to binary-valued.

3.2.5. *Produce the point around which constraints will be generated.* We now pick the random point p that will be maintained as feasible through the generation of constraints. We generate a vector of length `num_vars` from a Normal$(1000, 100)$ distribution. This ensures a point that has all positive coordinates.

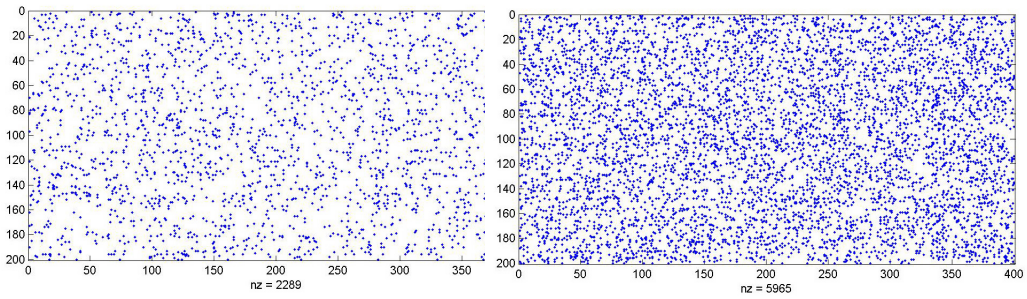
Next, we round every value that has been determined to be integer-valued. Approximately half will be rounded up, and half will be rounded down.

Next, we make all binary-valued variables equal to either zero or one. This is done randomly so that approximately half will be zero and half will be one.

The above two operations guarantee that the point will be feasible, in that it satisfies the binary- and integer-valued constraints.

3.2.6. *Generate constraints.* `num_constraints` tells us how many constraints must be generated. `density_range_hi` and `density_range_lo` specify a range of what percentage of the values in the matrix A are to be nonzero. We achieve this by generating each constraint according to the density ranges.

For each constraint, a density is chosen from Uniform(`density_range_lo`, `density_range_hi`) distribution. So if the range is $(0.01, 0.05)$, a density is chosen at random in this range. We now generate a vector of length `num_vars`



Low-density matrix. High-density matrix.

FIGURE 1. Illustrations of randomly generated matrices as shown using the `spy` function in MATLAB.

from Uniform$(0, 1)$ distribution. We will choose a coordinate of our constraint to be nonzero if the corresponding coordinate of this vector is less than the chosen density.

Now that we know which variables will have nonzero values in this constraint, we generate the actual values from a Normal$(400, 150)$ distribution. Thus, all of the non-zero entries in the matrix A are drawn from this distribution.

We continue this process `num_constraints` times, to generate all of the constraints. Graphical representations of low- and high-density matrices are shown in Figure 1. These were generated using the `spy` function of MATLAB, which creates a plot representation of sparse matrices. The x-axis represents columns in the matrix; the y-axis, rows. Points in the plot represent non-zero entries in the constraint matrix.

3.2.7. *Generate the right-hand-side (b) for each constraint.* We must now generate b (as in §1.2) such that the constraints are not binding for p, the point that should remain feasible. To do this, we calculate Ap, which gives us all the left-hand-side (lhs) values. For a constraint C, we set $b(C) = lhs(C) + \text{Normal}(500, 50)$. So we randomly add space between the constraint plane and the point that should remain feasible.

Note that the optimal feasible point could be relatively far from p.

3.2.8. *Clean up.* Several final steps are performed to make sure the problem is reasonable.

(1) In order to avoid degeneracies in the matrix A, ensure that each variable is involved in at least one constraint.
(2) In order to avoid unbounded problems, ensure that every variable is bounded above and below. Lower bounds are set to zero. Upper bounds are set to $p + \text{Normal}(2000, 100)$. This choice of mean and

Dimension	Values
Number of Variables	40, 60, 80, 100–1000 by 100
Number of Constraints	20, 25, 30, 35, 50, 75, 100, 125
Percent Integer	5%, 10%, 15%, 20%
Percent Binary	5%, 10%, 15%, 20%
Density	1–5%, 1–10%, 10–20%
Repetitions	10

TABLE 1. Parameters for generating synthetic problems in Series One and Two

standard deviation should create situations where the upper bound constraint is binding for some but not all variables.

(3) Round down the upper bounds for all integer variables. Set the upper bounds for all binary variables to 1.
(4) As a final check, we ensure that the relaxed problem (no binary- or integer-valued constraints) is solvable by our linear program solver.

4. Description of the experiment

We generated three different series of problems using the method described in Section 3.2.

4.1. **Series One: Tuning NCH.** Our current implementation of NCH has two parameters that can be specified at run time. (In fact, there are more, but they play no role in the present study.) To determine the operating characteristics of NCH with regard to these parameters, we created ten variants of NCH. These ten variants represent two levels of one parameter, and five levels of the other parameter. The first parameter takes on values of a or b, while the second takes on levels one through five. We indicate the variants of the NCH with designators such as "NCHa3", indicating that the first parameter is at level a, and the second parameter is at level 3.

We exercised the variants of NCH on Series One, a collection of synthetic problems. (See Section 6.1 for our results.) In this series, the number of variables ranged from 100 to 1000, and the number of constraints was 20, 25, 30, or 35. The density ranges for the constraint matrix were [.01–.05], [.01–.10], and [.10–.20]. The approximate fractions of variables that were binary and integral were each .05, .10, .15, and .20. (Note that in our study, the binary variables are considered to be separate from the integer variables, and do not form a subset.) For each combination of the above parameters, we generated 10 different problems that share the same parameters. See Table 1.

4.2. **Series Two: Comparing NCH against BAB.** We generated a second series of problems using the same parameters as in the first series, except that

the number of constraints was either 50, 75, 100, or 125. We applied BAB and selected variants of NCH to all of these problems. The results are in Section 6.2.

4.3. **Series Three: Large problems.** Since NCH had performed so well so far, we generated a series of larger problems. The number of variables now ranged from 1000 to 2000, and the number of constraints varied from 100 to 250. The other parameters remained the same as in the earlier series. We applied selected variants of NCH to these problems. Our purpose was not to compare NCH against BAB (since our experience suggested that BAB would take an inconveniently long time on many of these problems), but merely to see if NCH would continue to work as well as it had on earlier series. It did, as outlined in Section 6.3.

4.4. **MIPLIB 3.0.** We wanted to apply NCH to several recognized difficult problems in addition to our generated problems. MIPLIB 3.0 is a library of mixed integer programming problems assembled by researchers at Rice University to evaluate performance of MIP solvers. The problems were chosen because they are difficult, but not necessarily because they are representative of real-world problems. (The problems and their documentation can be found at `http://www.caam.rice.edu/~bixby/miplib`. See in particular the technical report by Bixby, Ceria, McZeal, and Savelsbergh.) While we believed that some MIPLIB problems (particularly those that are pure-integer or pure-binary) would not be amenable to our methods, we wanted to show that we could solve at least some of these difficult problems quickly. See Section 6.4 for details.

5. Metrics

The BAB method is well studied and can, given enough computational power, find the global optimal solution to a problem. However, the BAB method has several failings. First, it is designed to find the global optimum, so it may overlook feasible solutions that it finds early on. Or it may find an infeasible solution that is very close to a feasible solution, but overlook it. Furthermore, the time required to find a feasible solution, or the global optimum, varies a great deal. So the time to first feasible (TTFF) has a very high variance. As a practical matter, this variance can be problematic. For instance, if typical problems take anywhere from 1 second to 18 hours of time to solve, it is difficult to attempt solutions every day.

The problematic characteristics of the traditional BAB mentioned above motivate the metrics we used to compare NCH and BAB:

- Time to first feasible solution (TTFF). In practical situations, once a satisfactory feasible solution is found, finding an optimal solution is

	N	Min	Max	Mean	Std. Deviation
No. Constraints	3035	20	35	27.47	5.6
No. Variables	3035	40	500	210.68	161.9
Integral	3035	1	124	26.17	25.3
Binary	3035	1	114	23.04	22.5
Density	3035	.04	.193	.088	.045
TTFF					
BAB	3034	.04	620.20	6.9143	30.2
NCHa1	3011	.06	163.40	.5290	3.4
NCHa2	3028	.07	136.01	.4900	3.4
NCHa3	3031	.07	63.23	.2541	1.2
NCHa4	3028	.07	145.84	.3938	2.9
NCHa5	3030	.07	69.13	.2796	1.3
NCHb1	3026	.07	216.51	.5527	4.7
NCHb2	3029	.07	614.50	1.0548	15.7
NCHb3	3029	.07	489.85	.3956	8.9
NCHb4	3029	.07	217.30	.7456	6.7
NCHb5	3030	.07	511.60	.5033	9.3
QFF					
BAB	3034	.9795	1.0000	.9999	.0007
NCHa1	3011	.9491	1.0000	.9997	.0023
NCHa2	3028	.9712	1.0000	.9998	.0015
NCHa3	3031	.9712	1.0000	1.0000	.0005
NCHa4	3028	.9577	1.0000	.9998	.0017
NCHa5	3030	.9577	1.0000	.9998	.0014
NCHb1	3026	.8842	1.0000	.9998	.0027
NCHb2	3029	.8842	1.0000	.9999	.0022
NCHb3	3029	.8842	1.0000	.9999	.0021
NCHb4	3029	.8842	1.0000	.9998	.0026
NCHb5	3030	.8842	1.0000	.9998	.0025

TABLE 2. Series One: TTFF and QFF for BAB and Variants of NCH

often not enough of an improvement to be worth the effort. Indeed, the difficulty of finding a provable, globally optimal solution can be astronomical. For this metric, lower values are more desirable.

- Quality of first feasible solution (QFF). This is a ratio between the objective function values at the first feasible solution and at the relaxed optimal solution. For minimization problems where the relaxed optimal solution has a negative objective function value, QFF thus has a theoretical maximum of 1. For this metric, larger values are more desirable.
- Number of calls to our linear program solver. This measure is important, since it is independent of any possible difference in the relative

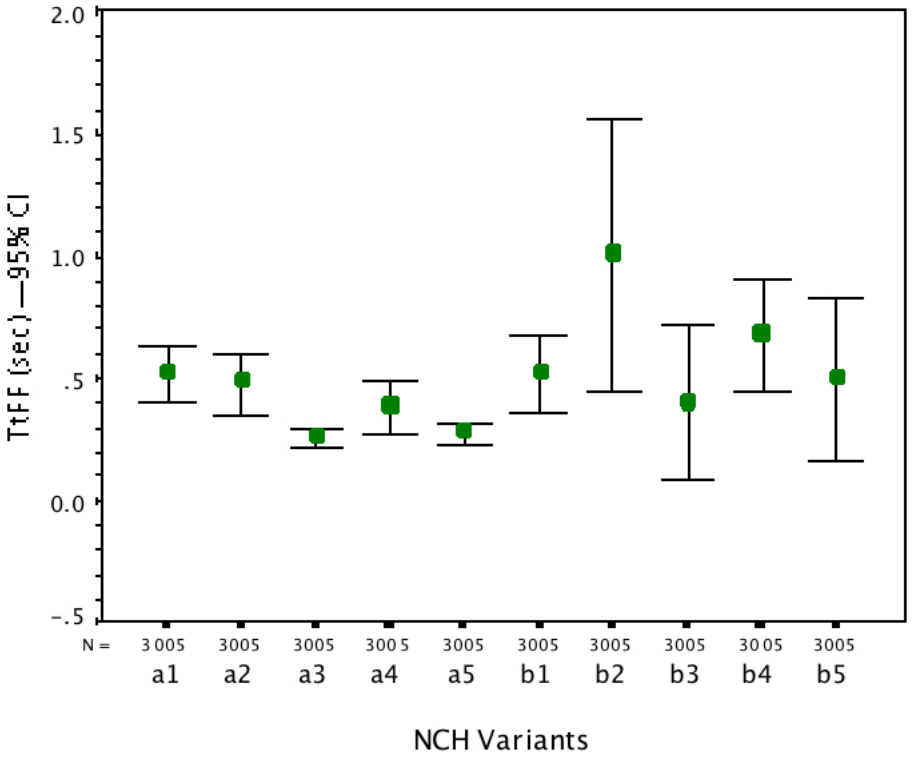


FIGURE 2. Series One: 95% Confidence Intervals of TTFF of NCH Variants

quality of our implementations of NCH and BAB. For this metric, lower values are more desirable.

6. Phase I Results

We accomplished our stated objectives (as listed in Section 2) and more.

6.1. **Results: Different variants of NCH.** We first ran ten NCH variants over the problems in Series One. Table 2 shows the mean time to first feasible (TTFF) and standard deviations, and the mean quality of the first feasible solution (QFF) expressed as a fraction of the relaxed optimal. The column with heading N represents the number of observations used to generate the statistics. The other columns, Min, Max, Mean, and Std. Deviation, represent the univariate statistics over the N observations. By inspection, NCH variants identified here as a3 and a5 consistently have the fastest times and highest solution quality. One other variant whose properties are nearly as attractive is b3. Figure 2 graphically represents the TTFF results in Table 2. The figure

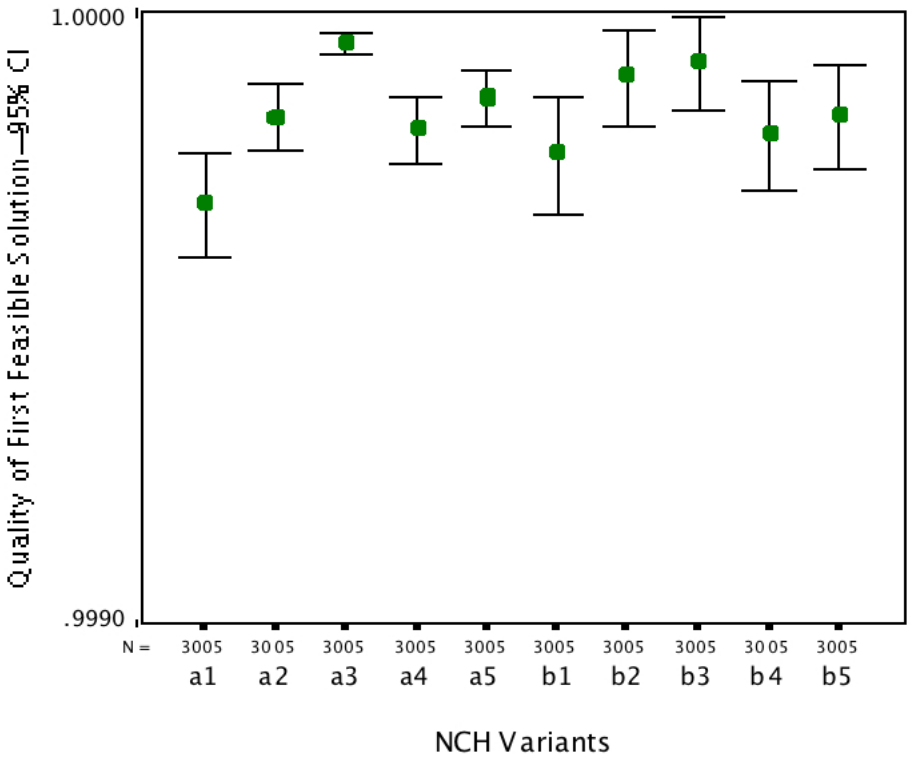


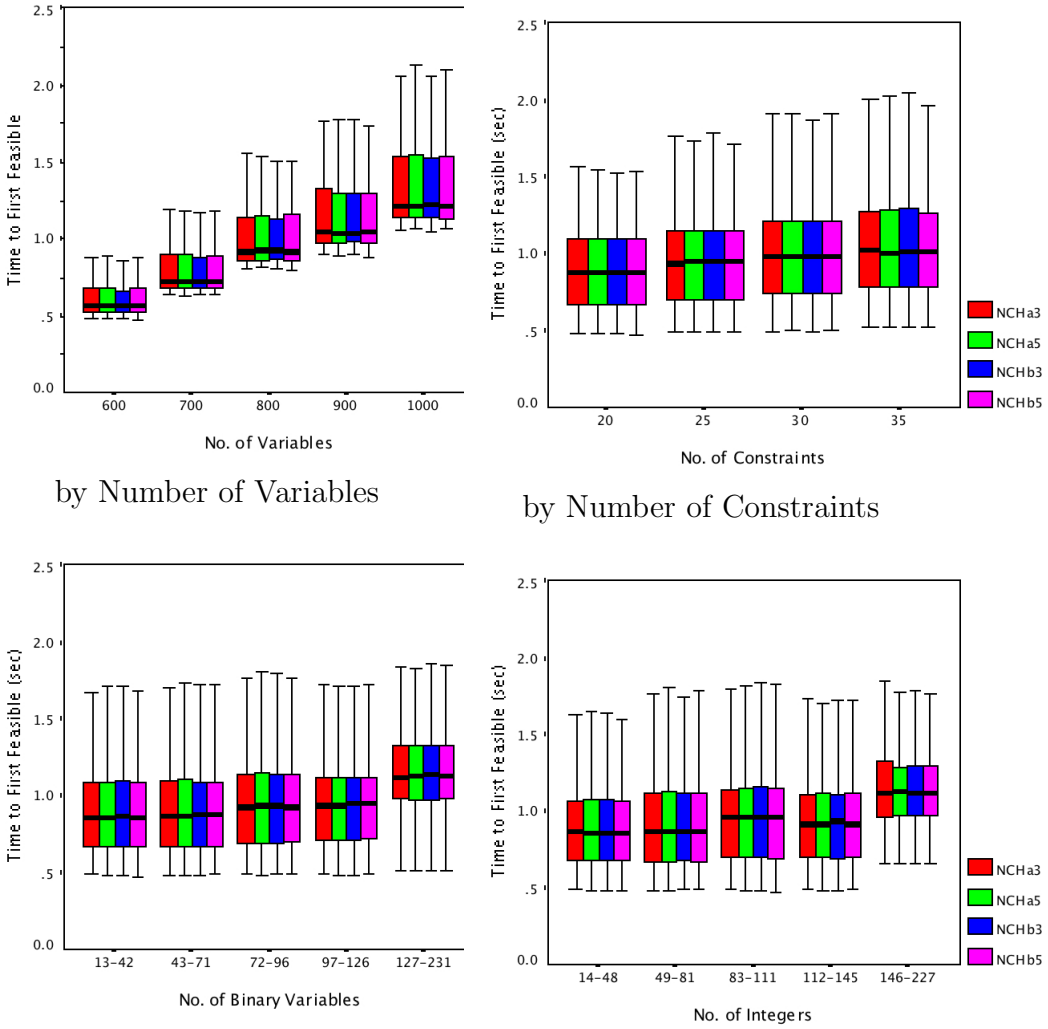
FIGURE 3. Series One: 95% Confidence Intervals of QFF of NCH Variants

shows 95% confidence intervals of the TTFF of the ten NCH variants. From the figure it is clear that NCHa3 and NCHa5 have low means and variances. These two variants stand out as finding feasible solutions quickly and consistently. Figure 3 shows confidence intervals for the quality of the first feasible solution (QFF) for all ten variants. As in the table, it can be seen that NCHa3 is the best.

To further explore the differences among the promising variants of the NCH identified above, we graphed boxplots of NCHa3, NCHa5, NCHb3, and NCHb5 by number of variables, number of constraints, number of binary variables, and number of integer variables. These graphs are shown in Figure 4.

A one-way ANOVA of these four variants comparing the mean times to first feasbile shows no significant difference between the four variants. See Table 3 (page 16).

Our main conclusion from these four plots is that the four variants are very comparable across these dimensions. In some sense, these four dimensions all represent proxies for the size, or difficulty, of the problem. So another



by Number of Variables

by Number of Constraints

by Number of Binary Variables

by Number of Integer Variables

FIGURE 4. Series One (second half): TTFF for NCH Variants a3, a5, b3, and b5

conclusion from these graphs is that the relation between the performance of the NCH and each of these dimensions is, at worst, linear. This can most clearly be seen with the upper left plot by number of variables.

6.2. **Results: NCH vs. BAB.** To understand the differences between the NCH variants and BAB, and the interactions between the dependent variables and the NCH parameters, we first examined the number of calls to the linear program solver. The number of calls to linprog is significantly smaller with

	Sum of Squares	df	Mean Square	F	Sig.
NCHa3 Between Groups	6.197	3	2.066	1.523	.206
Within Groups	4105.635	3027	1.356		
Total	4111.832	3030			
NCHa5 Between Groups	8.631	3	2.877	1.757	.153
Within Groups	4954.990	3026	1.637		
Total	4963.621	3029			
NCHb3 Between Groups	247.396	3	82.465	1.041	.373
Within Groups	239571.883	3025	79.197		
Total	239819.280	3028			
NCHb5 Between Groups	263.808	3	87.936	1.013	.386
Within Groups	262766.803	3026	86.836		
Total	263030.611	3029			

TABLE 3. One-way ANOVA: Comparison of Mean TTFF for fastest NCH variants

increases in each of the dependent variables. It can be seen that different dependent variables have corresponding different impacts on the number of linear program (LP) calls. Table 4 (page 17) and Figure 5 (page 19) show the number of LP calls graphed for the ten NCH variants and BAB across the different dimensions of our study. It is clear that BAB makes the most calls to the LP solver across all dimensions of the study. Furthermore, for BAB, there appears to be an exponential relationship between the number of LP calls and the number of integer variables (see the lower left plot). In contrast, for NCH, there is, at worst, a linear relationship between the various dimensions of the problems and the number of LP calls. This is an important result for NCH, since most of the computational effort comes from LP calls.

Using NCHa3, our fastest and most consistent variant, a systematic comparison was completed against BAB. The exponential impact of the number of variables, constraints, integers, and binaries can be clearly seen in Figure 6 (page 20). In general, NCHa3 is not subject to similar exponential effects. We examine each graph in turn.

The upper left-hand graph of Figure 6 displays boxplots of the TTFF for NCHa3 and BAB as the number of variables in the problems increases. This most clearly exemplifies the exponential relationship between the BAB's variance of TTFF and the size of the problem, i.e., the number of variables. In this plot, the outliers for BAB are not shown. Note the tight boxes for NCHa3. In general, on the problems in this series, the NCHa3 rarely takes more than several seconds, while the BAB could take from several seconds to over 1000 seconds to solve the same problems.

	N	Min	Max	Mean	Std. Deviation
BAB	3034	1	6181	81.63	333.86
NCHa1	3011	2	8842	9.61	164.62
NCHa2	3028	2	1746	7.49	55.54
NCHa3	3031	2	122	2.09	2.88
NCHa4	3028	2	710	4.86	30.81
NCHa5	3030	2	134	2.56	3.98
NCHb1	3026	2	2930	8.78	77.83
NCHb2	3029	2	8934	16.87	242.26
NCHb3	3029	2	962	2.38	17.64
NCHb4	3029	2	3568	13.34	140.21
NCHb5	3030	2	998	4.71	23.44

TABLE 4. Series One: Mean number of calls to MATLAB's linear program solver for NCH variants and BAB.

The upper right-hand graph of Figure 6 displays boxplots of the TTFF for NCHa3 and BAB as the number of constraints in the problems increases. Althought impact is less dramatic than with the number of variables, the basic pattern persists: BAB is subject to an exponential increase in processing time with an increase in the number of constraints, while NCH is impacted only linearly.

The lower left- and right-hand graphs of Figure 6 display boxplots of the TTFF for NCHa3 and BAB as the numbers of discrete variables, integer and binary respectively, in the problems increase. Here we can observe that the number of integer variables has a stronger impact on BAB than does binary variables, but neither of these parameters has a significant effect on NCH.

These results are supported by Multivariate Analysis of Variance. The dependent variable for the time to reach the first feasible solution was compared to the independent factors of the number of variables, constraints, integers, and binary variables. Note that the integer and binary variables are treated separately and binary variables are not a subset of integer variables. Results of the MANOVA are shown in Table 5 (page 18).

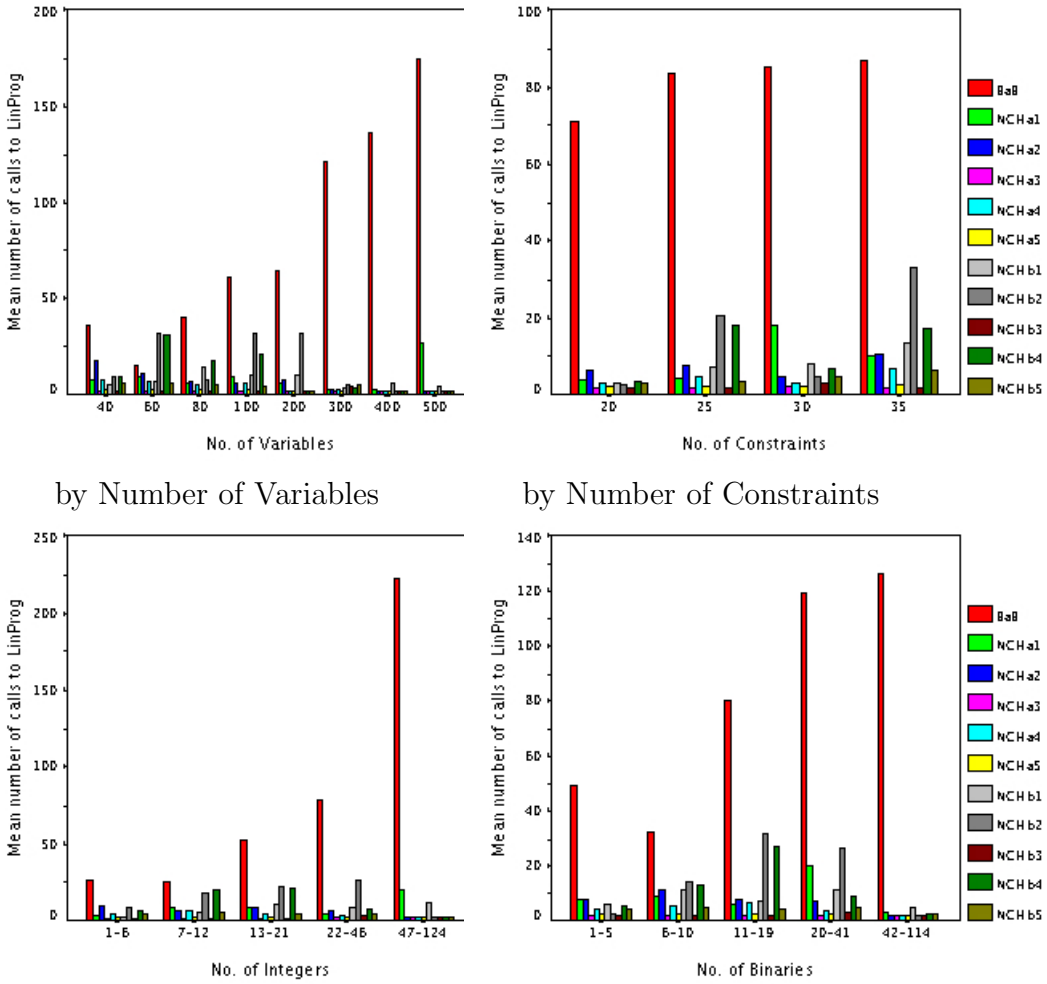
First note the corrected model is significant for both methods (BAB: $F = 1.70$, $p = .001$, NCH: $F = 4.99$, $p = .001$). This indicates the parameters explain a significant portion of variance for the amount of processing time of each method. Interestingly, the corrected R^2 for NCH is twice as large as that for BAB (.50 vs .26) further demonstrating that NCH times to a solution can be successfully and consistently predicted.

Looking at the individual factors, in can be seen that BAB has significant main effects for the number of constraints ($F = 4.20$, $p = .01$) and the number of integers ($F = 14.41$, $p = .001$). Clearly, the number of integers has the

Source	Dependent Variable	Type III Sum of Squares*	df	Mean Square	F	Sig.
Corrected	BAB	3143996	523	6011.465	1.698	.000
Model	NCHa3	2607	523	4.984	4.996	.000
Intercept	BAB	405127	1	405126.963	114.461	.000
	NCHa3	1214	1	1213.964	1217.003	.000
C	BAB	44607	3	14869.075	4.201	.006
	NCHa3	205	3	68.497	68.668	.000
V	BAB	5867	9	651.876	.184	.996
	NCHa3	169	9	18.746	18.793	.000
I	BAB	204037	4	51009.162	14.412	.000
	NCHa3	3	4	.752	.754	.556
B	BAB	8315	4	2078.723	.587	.672
	NCHa3	1	4	.263	.264	.901
C * V	BAB	28567	27	1058.044	.299	1.000
	NCHa3	34	27	1.254	1.258	.169
C * I	BAB	67184	12	5598.640	1.582	.090
	NCHa3	9	12	.769	.771	.682
V * I	BAB	73689	25	2947.552	.833	.702
	NCHa3	21	25	.827	.829	.708
C * V * I	BAB	187266	70	2675.227	.756	.934
	NCHa3	61	70	.868	.871	.769
C * B	BAB	12681	12	1056.785	.299	.990
	NCHa3	26	12	2.163	2.168	.011
V * B	BAB	41837	26	1609.114	.455	.992
	NCHa3	13	26	.484	.486	.987
C * V * B	BAB	169199	71	2383.078	.673	.984
	NCHa3	37	71	.525	.526	1.000
I * B	BAB	34911	16	2181.915	.616	.873
	NCHa3	10	16	.649	.651	.844
C * I * B	BAB	144532	42	3441.245	.972	.522
	NCHa3	28	42	.672	.674	.947
V * I * B	BAB	131847	61	2161.427	.611	.993
	NCHa3	37	61	.600	.601	.994
C * V * I * B	BAB	220760	137	1611.386	.455	1.000
	NCHa3	98	137	.719	.721	.993
Error	BAB	9096349	2570	3539.435		
	NCHa3	2564	2570	.998		
Total	BAB	14226232	3094			
	NCHa3	10627	3094			
Corrected	BAB	12240345	3093			
Total	NCHa3	5170	3093			

*BAB: R^2 = .257 (adjusted R^2 = .106).
NCH: R^2 = .504 (adjusted R^2 = .403).
C = No. of Constraints
V = No. of Variables
I = No. of Integral Variables
B = No. of Binary Variables

TABLE 5. MANOVAs for comparison of NCH and BAB



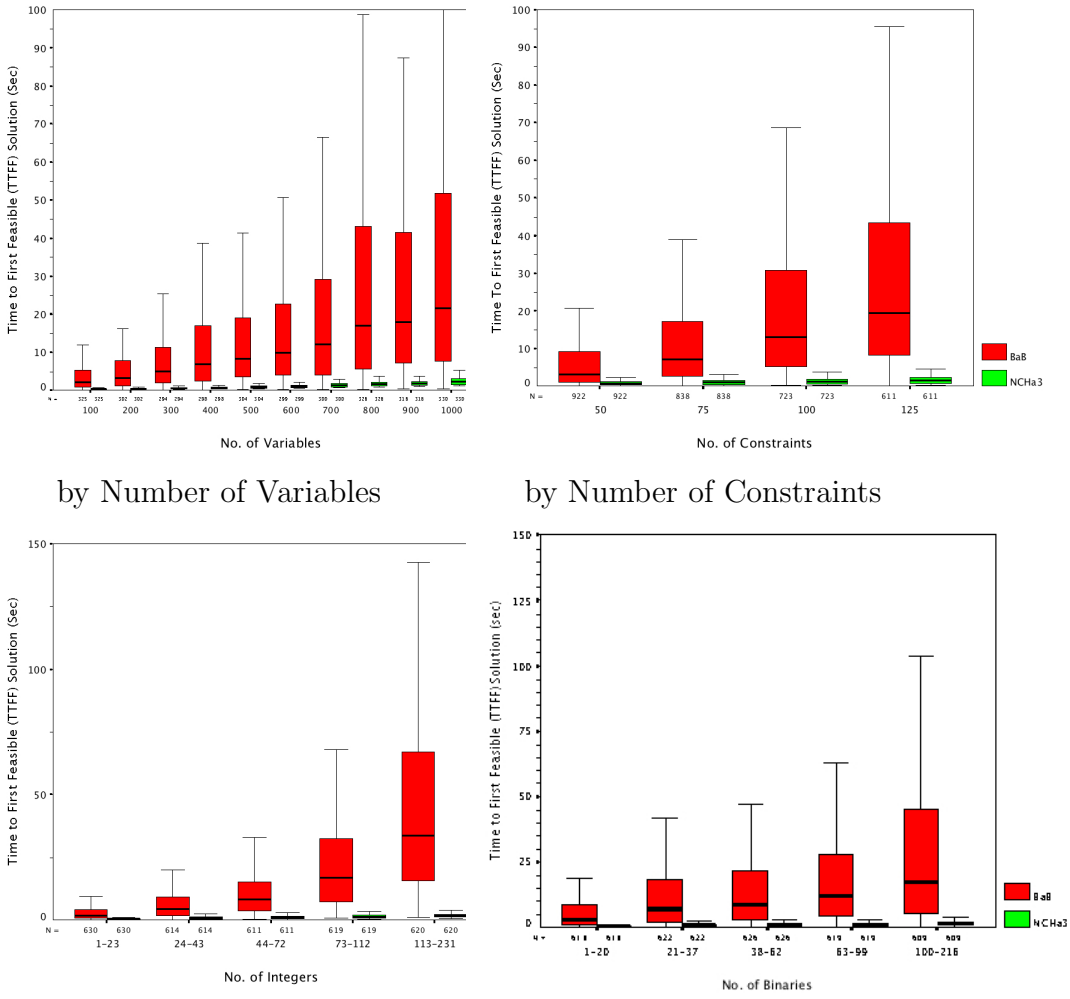
by Number of Variables by Number of Constraints

by Number of Integer Variables by Number of Binary Variables

FIGURE 5. Series One: Mean number of calls to MATLAB's linear program solver for NCH variants vs. BAB

major impact on increasing the amount of time to process problems using BAB. The parameters having a significant main effect on NCH are the number of variables ($F = 18.79$, $p = .001$) and the number of constraints ($F = 68.67$, $p = .001$). Further, the number of constraints interacts with the number of binary variables to impact processing time ($F = 2.17$, $p = .05$).

The statistics, tables, and graphics provide conclusive evidence that NCH processing time is not impacted by the number of integers in the problem. NCH is impacted by the number of variables and the number of constraints. However, it is predicately impacted and it appears in a linear, rather than exponential fashion.



by Number of Variables by Number of Constraints

by Number of Integer Variables by Number of Binary Variables

FIGURE 6. Series Two: TTFF, NCH vs. BAB

For a tabulated comparison of BAB and selected NCH variants, see Table 6.

To further explore the relationships of problem size and TTFF for NCH and BAB, we created scatter-plots of NCHa3 and BAB on linear and log scales in Figures 7 and 8 (pages 22 and 23), respectively. Due to the many outliers for BAB, it is difficult to compare the NCH and BAB on the linear scale graph. It is clear, however, that the variance of the TTFF for BAB increases with the number of integer variables. This is consistent with the previous literature findings for BAB. On the linear scale graph, note the NCHa3 TTFF points on the lower right edge of the graph. On the log-scale graph, the NCHa3 seems to

	N	Min	Max	Mean	Std. Deviation
No. Constraints	3101	50	125	83.29	27.5
No. Variables	3101	100	1000	554.31	291.2
Integral	3101	1	231	69.34	51.0
Binary	3101	1	216	60.52	44.9
Density	3101	.0260	.1663	.069502	.047
TTFF					
BAB	3100	.09	1048.73	25.2980	62.85
NCHa3	3095	.13	23.68	1.3277	1.29
NCHa5	3096	.13	23.63	1.3267	1.29
NCHb3	3097	.13	50.97	1.3218	1.48
NCHb5	3097	.13	51.05	1.3222	1.48
QFF					
BAB	3100	.9900	1.00	1.0000	.0004
NCHa3	3095	.7818	1.0000	.9999	.0039
NCHa5	3096	.7818	1.0000	.9999	.0039
NCHb3	3097	.7818	1.0000	.9999	.0039
NCHb5	3097	.7818	1.0000	.9999	.0039

TABLE 6. Series Two: BAB and selected NCH variants

level off to some extent as the number of integer variables increases. Clearly, the BAB is more dispersed and at a steeper angle.

While the mean time to achieve the first feasible solution (TTFF) shows a clear advantage in favor of NCH, it is important to understand the distribution of the outliers between the two approaches. Figure 9 (page 24) shows points outside of the 95% confidence interval for both BAB and NCH at various variable levels. Inspection of the graph shows that NCH outliers are of the same order of magnitude as the mean time. BAB outliers are at least an order of magnitude greater than the mean time. In the case of 1000 variables, the maximum time for BAB to solve is 1048 seconds (17.47 minutes) when the average time for NCH is 1.33 seconds and a maximum time of 23.7 seconds.

6.3. **Results: NCH on large problems.** While our selected NCH variants reliably performed better than BAB on the Series Two problems, we noticed that the larger the number of constraints relative to the number of variables, the more outliers NCH produced. For an example of such an outlier, look near the lower left of of Figure 7. While these outliers were an order of magnitude milder than BAB's outliers (60 vs. 1063), we still wanted to improve the situation. Thus, we enhanced NCH with a third tunable parameter, whose possible values were "off" and "on." When we refer to NCHa3, but with this parameter turned on, we call the resulting variant NCHa3+.

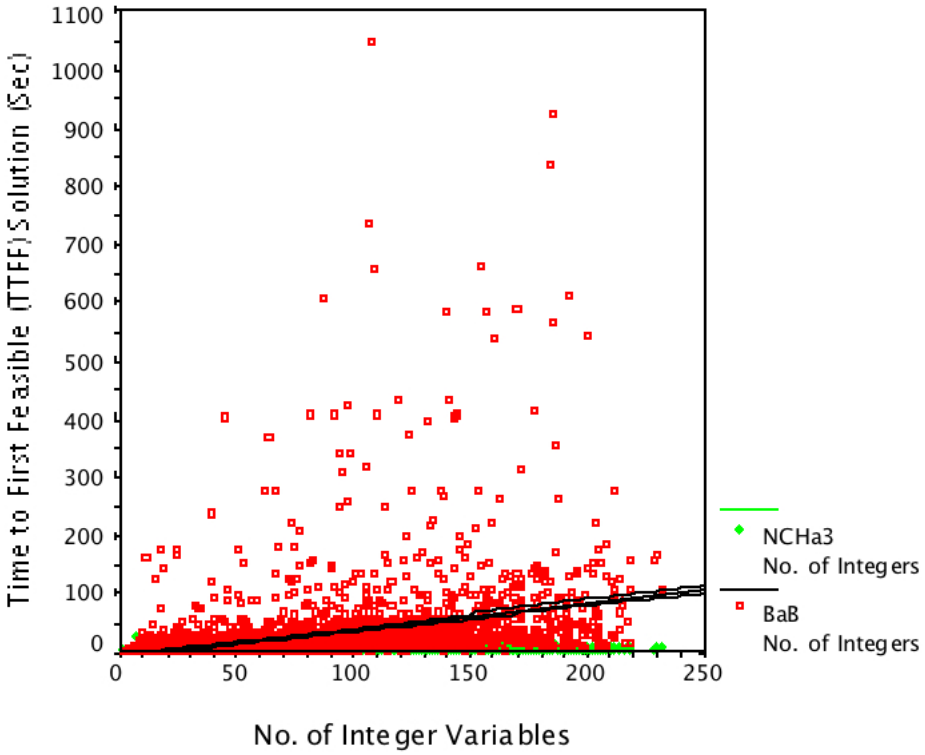


FIGURE 7. Series Two: Scatterplot of TTFF, NCH vs. BAB. Note that this figure shows a single NCH outlier at coordinates (10,25). This outlier is discussed in §6.3. The figure is truncated and BAB outliers have TTFF values greater than 1000 seconds.

To test whether NCH remained practical on larger problems, and to test the effect of this new parameter, we applied both NCHa3 and NCHa3+ to the problems in Series Three. The results are tabulated in Table 7. Confidence intervals for the mean TTFF and QFF appear in Figure 10 (page 24) and Figure 11 (page 11). Note that, as we hoped, NCHa3+ was as good as NCHa3 in general, but had fewer and milder outliers when the number of constraints was approximately the same magnitude as the number of variables.

6.4. **Results: NCH on some known difficult problems.** NCHa3 was able to solve several MIPLIB problems quickly. The results are presented in Table 8 (page 8). The problems `gesa2_o` and `gesa3_o` come from a real-world application: optimizing electricity generation in the Balearic Islands of Spain. The variants `gesa2` and `gesa3` are the same problems, but slightly reformulated.

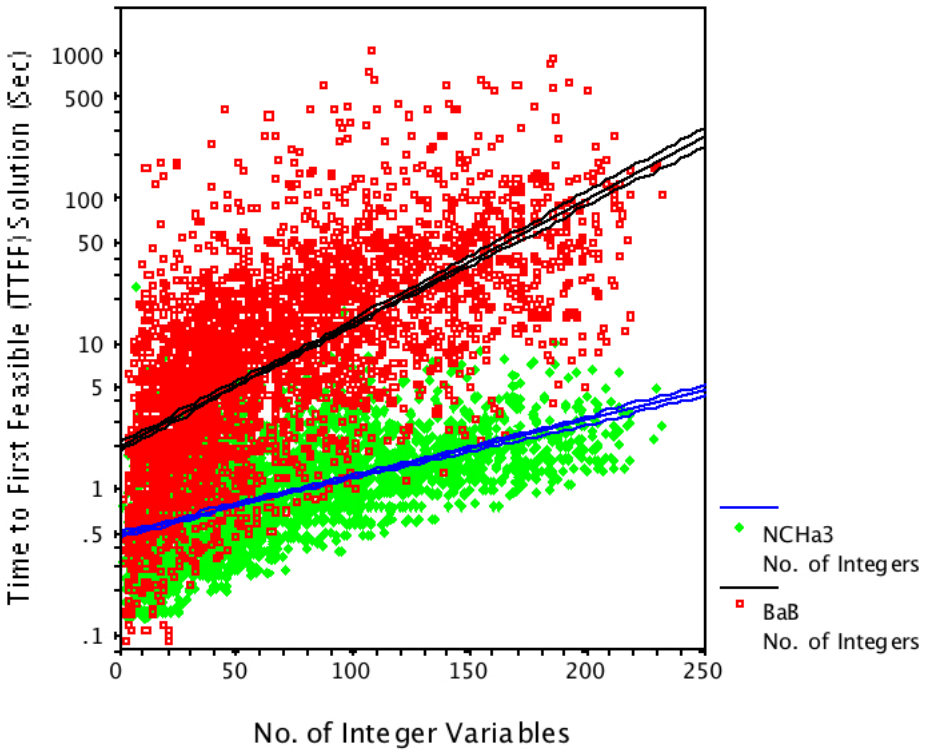


FIGURE 8. Series Two: Log Scale Scatterplot of TTFF, NCH vs. BAB

	N	Min	Max	Mean	Std. Deviation
No. Constraints	427	150	250	185.36	39.2
No. Variables	427	500	2000	1318.50	575.7
Integral	427	20	440	171.91	109.5
Binary	427	15	421	141.70	94.2
Density	427	.0275	.1529	.0458	.028
TTFF					
NCHa3	426	1.11	55.19	9.6135	7.6
NCHa3+	426	1.09	53.36	9.5348	7.3
QFF					
NCHa3	426	.999928	1.000000	1.0000	.0000
NCHa3+	426	.999985	1.000000	1.0000	.0000

TABLE 7. NCH on Large Problems: Mean TTFF, QFF, and standard deviations for NCHa3 and NCHa3+

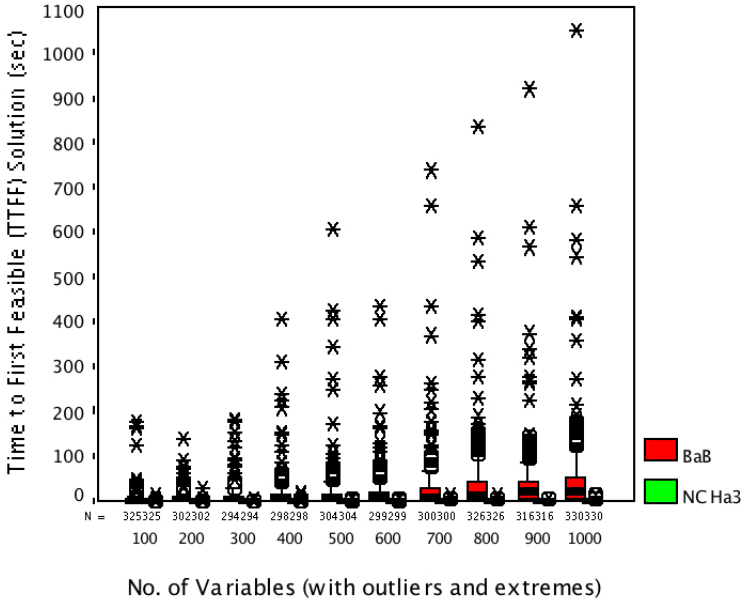


FIGURE 9. Series Two: Boxplot of Mean TTFF by Number of Variables with Outliers and Extremes

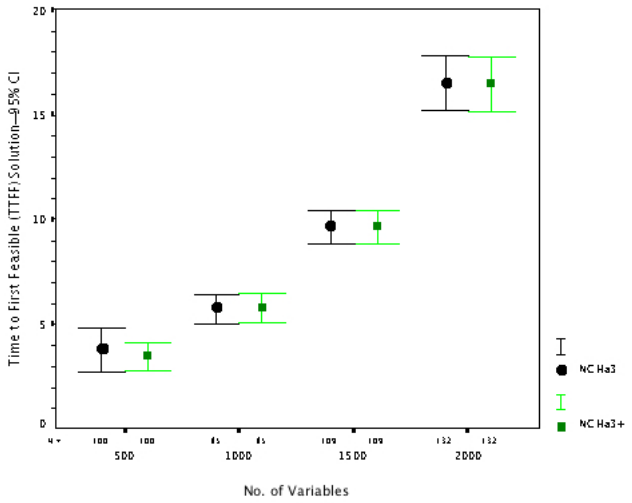


FIGURE 10. NCHa3 and NCHa3+ on Large Problems: Error Bar Plot of Mean TTFF with 95% Confidence Interval

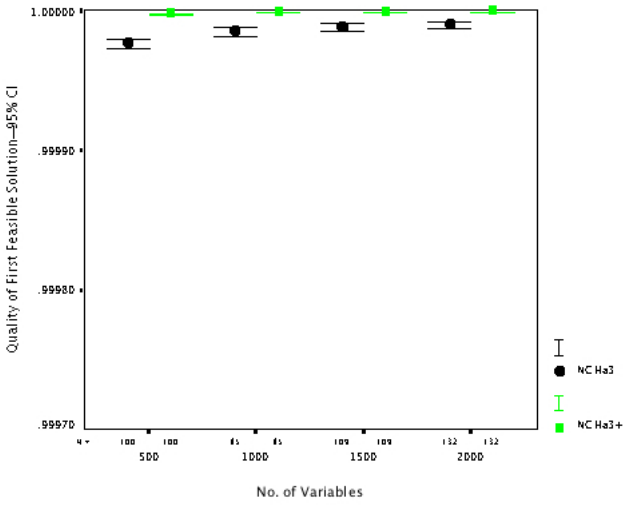


FIGURE 11. NCH on Large Problems: Error Bar Plot of Mean QFF with 95% Confidence Interval

					NCH		BAB	
Name	Variables	Constraints	Integer	Binary	TTFF (sec)	QFF	TTFF	QFF
gesa2	1392	1224	168	240	12.09	1.007	65.17	1.042
gesa2_o	1248	1224	336	384	16.97	1.007	41.43	1.045
gesa3	1368	1152	168	216	11.02	1.003	29.96	1.024
gesa3_o	1224	1152	336	336	15.27	1.003	20.39	1.023

TABLE 8. NCHa3+ and BAB on some MIPLIB problems. Note that for these problems, QFF has a theoretical *minimum* of 1.

6.5. **Summary.** The Phase I objectives were met and the success of the heuristic was demonstrated.

First, when other parameters are fixed, the number of integer and binary variables in a random MIP problem does *not* have an exponential impact on the time required to find a feasible solution using NCH. Second, the performance data show that NCH produces feasible solutions significantly faster than BAB. Moreover the variance in time to produce a feasible solution is smaller in NCH.

This reduction in variance and the resulting greater predictability of time to first solution will be extremely attractive to logistic companies and consultants. These properties of NCH are very exciting in terms of optimization in general.

The heuristic is ready to be lifted from the MATLAB testbed and moved to a software production platform and integrated into what will become the

Adaptive Mixed Integer Problem Solver (AMIPS) software. It is this software package that is the focus of Phase II development.

7. Further Potential Speed Enhancements to NCH

7.1. **The adaptive potential of NCH.** NCH has attractive features that are uniquely suited to the requirements of the Navy Comprehensive Optimal Manpower and Personnel Analytic Support System (COMPASS) project. Within the architecture of the COMPASS program is an optimization module. First, the COMPASS simulator is seeded with a collection of feasible solutions from which simulations are derived. NCH is uniquely able to quickly produce a specified number of feasible solutions that are scattered in the feasible space. This is a property that is not available directly from any current optimization package. Second, the COMPASS simulator produces a set of parameters that would allow information to produce better solutions to be passed to the optimizer. In a feedback scenario, where the optimization module is called repeatedly on similar problems, and particularly in highly parallel installations, we can exploit the extra knowledge available to adapt at run time. Of course, pursuing this advantage will require direct engineering collaboration between the contractors.

7.2. **The Potential of NCH as a parallel algorithm.** NCH is *embarrassingly parallelizable*, meaning that the algorithm is naturally adaptable to a parallel architechture. In particular, under NCH, one could parcel out pieces of the problem to many processors, which would have only a limited need to communicate with each other.

8. Phase II Objectives

In Phase II our major objective is to deliver a software production platform that will host a commercial-quality adaptive mixed-integer programming solver (AMIPS). The AMIPS will be based on the successful heuristic demonstrated in Phase I. Our business model is to deliver optimization services directly to businesses and through consultants who use our services after problem formulation activities.

Phase II will be carried out in two stages, each focused on a key deliverable. Stage one will produce the commercial quality AMIPS software. Stage one activities will include four major tasks: growing and perfecting the algorithm; rehosting our solution onto our production platform; parallelizing the platform to support distributed computing; and building the XML and Web interfaces necessary to pass appropriately formatted data into and out of the solver platform.

Stage two (the Phase II option) will focus on integrating the AMIPS into the COMPASS simulator as the optimization module. Stage two will consist

of three major tasks. The software will be installed within the COMPASS simulator hardware environment. An application program interface will be created to pass information between the AMIPS optimization module and the COMPASS simulation module. An expert system will be created to receive and utilize information from the simulator and modify and tune the search for feasible solutions.

8.1. **Objective 1: The AMIPS Software Production Platform.** At the end of Phase I, the NCH operates as a routine within the MATLAB programming environment. The programming code must be moved onto a production software platform. The first task is to grow and perfect the algorithm, to expand the robustness of NCH and improve its performance on a wide range of problems. This will require the creation of several new software modules including several preprocessing modules and a linear programming solver. The most important is the linear programming module. In order to achieve the control and transparency required to diagnose and eliminate problems, our first effort will be to evaluate commercial and existing open-source implementations of linear programming solvers. At this time it our intention to customize an open-source linear program solver and incorporate it into our software package. We estimate this will take 3 months to complete and will be accomplished under the supervision of Dr. Adler. Standard optimization techniques used for preparing data for analysis in BAB techniques will benefit NCH as well. Drs. Adler and Kline will develop the preprocessing, cutting methods, cycle detection and correction routines, and modules to support specialized constraint types. This effort will be assisted by Creative Action Program Director Scott Collins and a senior programmer. This effort will take approximately 4 months and will be done in parallel with the linear programming module development.

The second task of stage one will be to rehost our solution onto the production software platform. The programming team will unify the software modules into a working application. The application will have the necessary application programming interfaces to read in and pass out standard matrix formats. This effort is expected to take approximately 2 months and will be headed by the Creative Action programming staff.

The third task will be to parallelize the platform and maximize the distribution capabilities of AMIPS. This effort will take approximately 1 month and will be headed by the Creative Action programming staff.

The fourth task will integrate the software production platform to provide the necessary capability to support off-site web services. This will involve the development of a toolkit to build customer interfaces and the corresponding

XML and Web interfaces. Training materials and help documentation will be completed to provide the necessary customer support.

At the end of Year 1, a commercial adaptive mixed integer program solver will be ready to provide logistics customers with optimization services and to be integrated for use in COMPASS and other Navy optimization requirements.

8.2. **Objective 2: Integration with the COMPASS Platform.** As our Phase II option, Creative Action in conjuction with our Research Institution partner, The University of Akron, will work in collaboration with the COMPASS contractor to integrate AMIPS as the optimization module in COMPASS. Stage two will consist of three major tasks. The first task will be the initial test of our enterprise solution. The AMIPS hardware, a multinode clustered computer installation, will be networked with the COMPASS simulator hardware. Because of the unique, adaptive capabilities of the COMPASS simulator, a customized application program interface must be created to pass information between the AMIPS optimization module and the COMPASS simulation module. Once the two systems are successfully exchanging information, work will turn to enhancing an expert system to receive and utilize information from the simulator. The intelligence provided by the simulator will be used to modify and tune searches for feasible solutions. It is expected that over time, better simulations will be produced by COMPASS, and those solutions will be produced faster, in part with the aid of the AMIPS optimization.

9. Transition Plan

Creative Action is currently developing an adaptive solver for optimization problems that dramatically reduces the time and variance in solving the hardest optimization problems. Our solver is based on our proprietary Neighborhood Covering Heuristic. The project is a joint effort with the University of Akron, Ann Arbor Digital Arts and Dr. Douglas Kline of the University of North Carolina, Wilmington.

9.1. **Customer Need.** The problems which the AMIPS system can solve are faced by every large business, or business that manages large inventories, significant human resources, capital-intensive construction, maintenance, or investment. AMIPS aids inventory management, airline scheduling, Internet/Intranet management, network routing, telecommunications, and anything that maximizes product profitability through faster and more predictable processing. The types of models for which our company will provide improved solution capability are used by most large corporations as well as the military. In addition, organizational consultants who specialize in this area may be potential resellers of our service.

The data gathering, problem formulation, and matrix generation differ for each business and problem type. In order to transition from the development stage to a commercial entity, we will focus on human resource management optimization, working in partnership with COMPASS contractors in order to provide the unique feasibile solution and adative qualities required. We will also support a transportation optimization effort that would support several logistic consulting companies who have expressed an interest in our software product and our off-site optimization services.

Each potential problem is large, has unique parameters, and is computationally expensive using current MIP solutions. Our vision is to create a customizable interface using an open source web browser development environment. The browser window is the control interface by which a customer can upload and solve a large-scale mixed-integer programming problem from the source-platform of their choice. Templates will be constructed for particular business sectors. We will develop the basic architecture and core templates through Phase II SBIR funding.

9.2. **Partnering.** Lewis & Clark Ventures LLC (LCV), an investment capital fund, has agreed to participate to help commercialize AMIPS. The fund focuses on investments in sectors including life sciences, micro- and nanotechnology, advanced material science, fuel cell technology and information technology. Fund manager Dr. Ron Clark, principal of Lewis and Clark, is the former president and executive director of the Ohio Polymer Enterprise Development Corp., a non-profit group created by the University of Akron in Ohio that provided upward of $25 million in VC funding for 30 Ohio-based startups. His partner in LCV is Robert Acri, a principal of Kenilworth Asset Management LLC, a money management firm based in Chicago. LCV will make investments that range from a $500,000 to $1 million over several funding rounds in each startup it backs, though in some instances, the total investment in a startup could be as large as $5 million. The fund expects to see early profit on many of its investments and aims to offer investors a 28% annual return.

Dr. Peter Weinstein, the Project Manager for Altarum on the Navy Comprehensive Optimal Manpower and Personnel Analytic Support System (COMPASS) project, has written a letter of support to Creative Action. Altarum would welcome the opportunity to work more closely with Creative Action LLC, Ann Arbor Digital Arts, and the academic members of our team. Based on the schedules presented by both the projects, there is the opportunity for joint development to the benefit of both projects.

10. Phase III Activities

10.1. **Funding Strategies.** With the support of our commercialization partner, LCV, we have already begun efforts to develop customer interest and have identified interested investment sources. Our business model provides for three ways to access our solution engine. A customer can engage our company to solve the problem directly as a consultant. Access could also be provided by subscription for a monthly or annual fee for customers who would have customizable templates they could access securely over the Internet. Finally, for large customers interested in maintaining absolutely independent data, particularly government and military customers, we would sell an enterprise solution that would include a license for the software, set up and maintenance of equipment, software updates, and technical assistance on-call for their entirely on-site instantiation. We anticipate that at the end of year 1, we will be able to work with a small set of interested logistics consultants to provide optimization solutions to supplement development costs, build the customized web interfaces, and tune the optimizer for targeted optimization problems. These include the Thorndyke Group, Altarum, and Sage Tree, a logistics software consultant.

In February, the Ohio Department of Development plans to announce the ODOD Commercialization Grant. Creative Action, as a Tibbet's Award recipient for excellence in contributions to and from the SBIR program is a likely candidate for these supporting funds. The funds are intended to aid in marketing and manufacturing efforts disallowed by STTR funding. These funds are targeted at companies with on-going Phase II projects.

Should additional development funds be required to expand the capabilities and infrastruction of our planned web-based off-site optimization service, Lewis and Clark Ventures has agreed to provide or solicit seed funding. It is anticipated that the Phase II funds will be sufficient to provide a working AMIPS production platform and to provide optimization services to the Navy and to other business customers at the completion of Phase II. In order to support the COMPASS effort, additional specialized developments in conjunction with the COMPASS primary contractor will be required, as noted above.

10.2. **Company Strategy and Intellectual Property.** The intellectual property agreement signed by Creative Action LLC, The University of Akron, Ann Arbor Digital Arts, and Douglas Kline, Ph.D. stipulates that a holding company directed by Creative Action will be created. See Figure **??**. The holding company, AMIPS Holding LLC, will hold exclusive license to all the intellectual property relative to the STTR research effort and future development by the holding company partners. With continued project success, Lewis

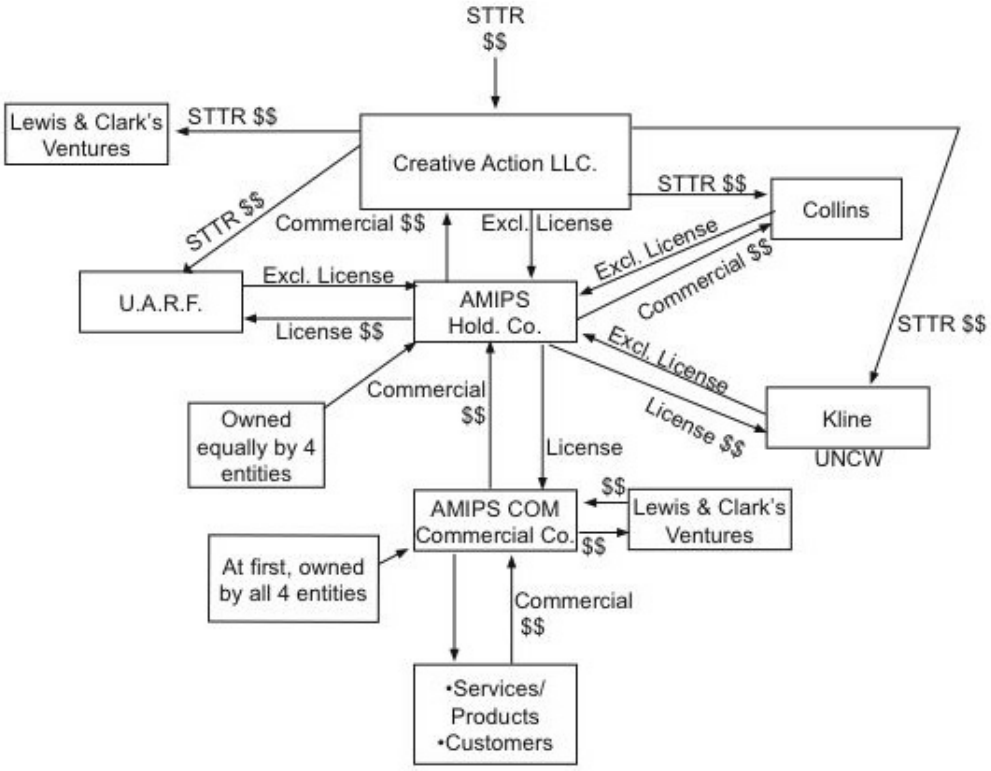


FIGURE 12. Structure of Intellectual Property Agreement Including the Holding Company and Funds Distribution.

& Clark Ventures is interested in supporting the funding of AMIPS Commercial LLC, to produce and distribute optimization installations, software, and services to businesses and military customers.

Although this will be the first commercial product Creative Action and its research partners have developed for the Department of the Navy, Creative Action has received four Phase II SBIR grants from the National Institute of Health and has produced five patents, maintains six trademarked products, and has two commercial projects on the market, with a third almost ready to go to market. These products include a system for communicating with foreign-language-speaking long-term care residents, and Medication Reminder software designed for use by adults and specifically older adults using a personal digital assistant.

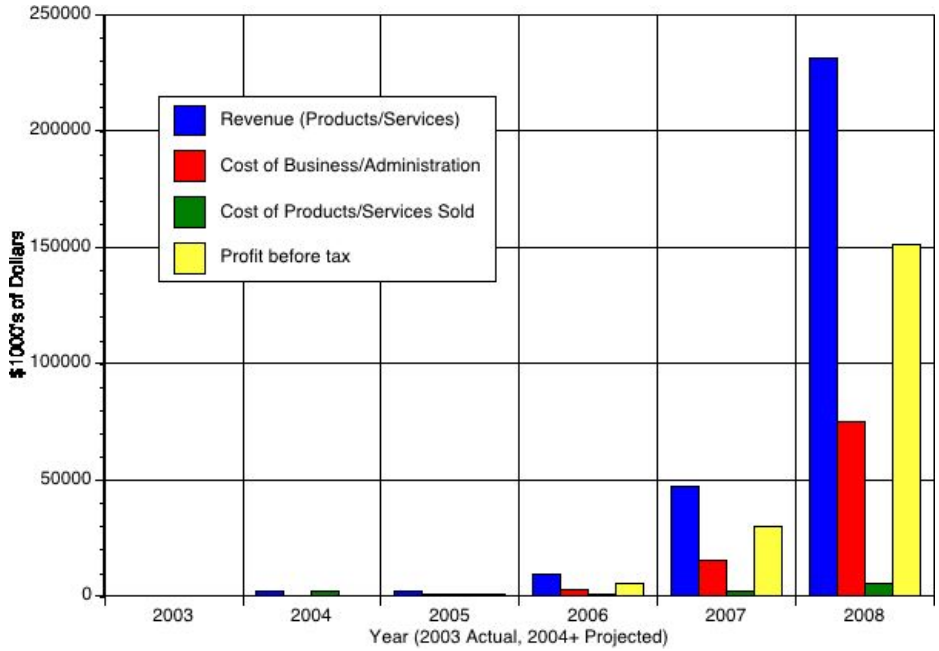


FIGURE 13. Financial potential of the AMIPS commercialization effort. The Figure assumes Phase II funding in 2004 and 2005 with positive revenue through 2008.

At the completion of the Phase II funding, Creative Action will have completed the development of the production software platform. The Phase II option will focus on the installation and integration of the software into the Navy Comprehensive Optimal Manpower and Personnel Analytic Support System (COMPASS) project. We have already begun a relationship with Altarum and CSC and anticipate that a joint development effort can be accomplished. We will supply the necessary technical and engineering support to deliver the Adaptive Mixed Integer Program Solver (AMIPS) technology either directly, or through a license with the primary COMPASS contractor.

10.3. **Market Potential.** We will work with logistics consultants to present our product to their existing clients. Each service level (consulting, subscription, and enterprise) would have a corresponding fee. While we expect to come up with a more sophisticated pricing structure, we have used the figures below as the basis for the financial projectsions in Figure 13.

We estimate that in the first 3 years, 15 Fortune 1000 corporations would use our services. Our consulting fee for each service would be $20,000. We expect another 20 Fortune 1000 corporations will pay our annual fee of $50,000 to access this unique service that has the potential to save them millions of dollars through timely data-driven solutions. Our enterprise solution will be

sold for $200,000 including our certified hardware that will be comprised of an array of blade processors, specifically configured for parallel computation, possibly as a Beowulf cluster. Additional revenue, $65,000 per year, will be generated through a support agreement that includes standard development training, maintenance, and updates to the software. We anticipate the Navy to be our first enterprise customer with two additional customers in the first year. This would provide the basis for the Navy to transition to a web-based marketplace for matching human resources to jobs. In each of the cases the customer is provided with a powerful customizable solution engine with tremendous computational power accessible to the extent the customer specifies.

References

[1] Bixby, Robert E., Cassandra M. McZeal, Sebastian Ceria, and Martin W. P. Savelsbergh (1998), *An Updated Mixed Integer Programming Library: MIPLIB 3*, Rice University Dept. of Computational and Applied Mathematics Technical Report TR98-03, `http://www.caam.rice.edu`.

[2] Glover, Fred W., and Manual Laguna (1997), *Tabu Search*, Kluwer Academic Publishers.

[3] Goldberg, David E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley.

[4] Goldberg, David E. (2002), *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Kluwer Academic Publishers.

[5] Knjazew, Dimitri (2001), "OmeGA: A competent genetic algorithm for solving permutation and scheduling problems," *Genetic Algorithms and Evolutionary Computation*, Vol. 6, Kluwer Academic Publishers.

[6] Nemhauser, George L., and Laurence A. Wolsey (1988) *Integer and Combinatorial Optimization*, Wiley, New York.

[7] Metrowerks, *CodeWarrior*, `http://www.metrowerks.com`

[8] Mathworks, *MATLAB*, `http://www.mathworks.com`

[9] SPSS Corporation, *SPSS*, `http://www.spss.com`

[10] Winston, Wayne L. (1997), *Operations Research: Applications and Algorithms*, 3rd ed., Duxbury Press.